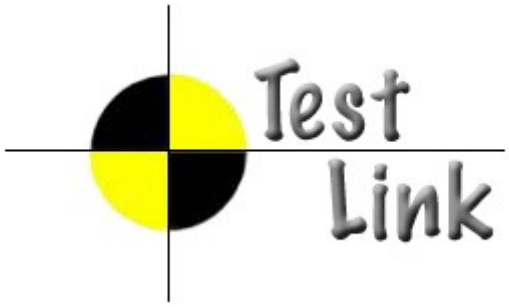


Installation & Configuration Manual

TestLink version 1.7

1. Scope

This document serves as a reference and knowledge base for the installation and configuration of tool **TestLink 1.7**. The first part includes the installation procedure and second part the configuration explanation.

The latest documentation is available on TestLink homepage. You can also ask a help to solve your problems in an appropriate section of TestLink forum.

Summary of installation process:

1. Install background services
2. Tranfer and uncompress files into web directory
3. Generate database tables and add data (create default or transfer from previous db)
4. Edit configuration files
5. PHP File extensions
6. Login

TestLink includes installation scripts that helps you easily setup all required configuration and database structure.

Table of Contents

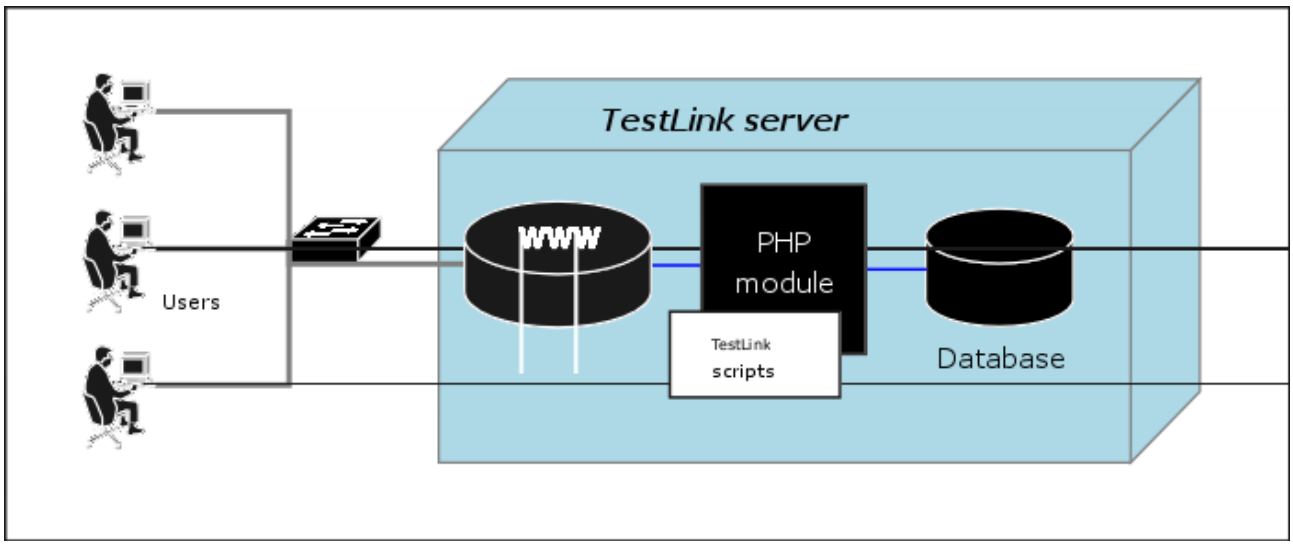
2. System Requirements

We support www browsers **Firefox** 1.0 (and higher) and **Internet Explorer 6**. There are some issues with IE 7 because microsoft doesn't satisfy standards. Generally any other browser should work if it suports javascript, XHTML and CSS.

TestLink server requires these applications as background:

- **Database**
  - MySQL 4.1.x and higher (4.0.x doesn't support UTF-8)
  - Postgres 8.x and higher
  - MS SQL 2000 and higher (experimental)
- **php** 5.x and higher (at least version 5.2 is recommended)
- **Webserver** (Apache 1.3.x or 2.x and higher, IIS 3 and higher, etc.). See `<php_root>/install.txt` for more information.
- **Bug tracking system** (optional)
  - Bugzilla 0.19.1 and higher
  - Mantis 1.0.1 and higher
  - JIRA 3.1.1 and higher
  - TrackPlus 3.3 and higher
  - Eventum 2.0 and higher
  - Trac 0.10 and higher
- There is no requirement about your operating system (tested on Linux and MS Win32).

You can run also your database on different server than TestLink php scripts.


TestLink server
Users
WWW
PHP
module
TestLink
scripts
Database

3. Installation

You can use automatic scripted installation or manual steps. If you are upgrading from a previous version of TestLink look at the Upgrading section.

3.1. Pre-installation steps

Do the next steps before installation:

1. Install environment: **Webserver with php5** and **database** (MySQL or Postgres). Refer to documentation of these products. You can also find installations package of all these products and install it together; for example XAMPP, EasyPHP, Uniform Server, etc.

   PHP4 is not supported from TL 1.7 version

2. Transfer the TestLink installation file to your webserver using whatever method you like best (ftp, scp, etc.). You will need to telnet/ssh acces into the server machine for the next steps (if not localhost).

3. Next, untar/gunzip it to the directory that you want. The usual command is (1 step):

   ```
   # tar zxvf <filename.tar.gz>
   ```

   or

   ```
   # gunzip <filename.tar.gz>
   # tar xvf <filename.tar>
   ```

   Winzip, Stuffit, and other programs should also be able to handle decompression of the archive.

   At this point you may want to rename the directory to something simpler like 'testlink'. You will use the mv command to rename a directory (Windows users substitute the "ren" command or use explorer).

   ```
   # mv <directory_name> testlink
   ```

4. Continue Installation or Upgrade.

3.2. AUTOMATIC Installation

TestLink includes installation scripts that helps you setup all required configuration and database structure. The following details the basic steps for installation on any system. The instructions may seem unix-centric but should work fine on Windows systems. Barring complications, it should take you about 10-30 minutes to install, configure, and be using TestLink.

This installation process has changed with release 1.6. Next we will create the necessary database tables and a basic configuration file.

1. From your web browser access **`http://<yoursite>/testlink/install/index.php`**.

2. This page will walk through the following steps:

   - check basic parameters for the web server, php config and DB version.

- prompt for the database type and location, and a database user/password pair. For installation, an administrative user/password pair can also be provided. The operating user requires ALTER, SELECT, INSERT, and UPDATE privileges. For installation, INDEX, CREATE, DELETE, and DROP privileges are also required.
- create the database and tables.

  Warning: *A DEFAULT ADMINISTRATOR level account is created.*

  The account name and password are: **admin / admin**. Use this when you first login to TestLink. Immediately go to Manage and create at least one administrator level account. You can recreate it but you should delete the account to prevent the cookie_string from being used to trick the package. It would be even better to rename the account or delete it permanently.

  SECURITY: *After setting up the package, remove the default admin account*
- perform some post installation checks on the system.

3. After a successful upgrade you should remove the `<testlinkwebdir>/install/` directory for security reasons.
4. The next part involves configuring the installation to work with your specific setup. See configuration section for description of configurable parameter.

3.3. MANUAL Installation

If you want to perform a Manual installation (**not recomended**) here are the steps needed for a successful installation. For installing the DB you can either choose the command line tools available in your MySQL installation or any MySQL Database Client (e.g. phpMyAdmin).

- Prepare MySQL via command line tools:
  - Create a new empty MySQL database.

    for MySQL >= 4.1 (with UTF8) do **CREATE DATABASE testlink CHARACTER SET utf8 COLLATE utf8_general_ci** By choosing UTF8 you should also change the value of DB_SUPPORTS_UTF8 to TRUE in your `<testlinkdir>/config.inc.php` See Configuration for more.
  - Create tables for the newly created database.

    ```
    # mysql -u <user> -p<password> <dbname> <
    <testlinkdir>/install/sql/testlink_create_tables.sql
    E.g. # mysql -u testlink -ppass testlink <
    /var/www/html/testlink/install/sql/testlink_create_tables.sql
    ```
  - Populate initial data for the newly created database (admin account, default roles).

    ```
    # mysql -u <user> -p<password> <dbname> <
    <testlinkdir>/install/sql/testlink_create_default_data.sql
    ```
- Alternatively you can use phpMyAdmin:
  - Create new database from main page (recommended UTF-8 character set).
  - Optionally create a new user and assign him correct rights for the created

database.

- Select the created database in the left pane.
- Navigate to SQL window.
- Upload SQL request from files `/install/sql/testlink_create_tables.sql` and run the script.
- Upload SQL request from files `/install/sql/testlink_create_default_data.sql` and run the script.

- Create a <testlinkdir>/config_db.inc.php file with the following data (example):

```
<?php // Automatically Generated by TestLink Installer
define('DB_TYPE', 'mysql');
define('DB_USER', 'testlinker');
define('DB_PASS', 'testlink_pass');
define('DB_HOST', 'localhost');
define('DB_NAME', 'tl_master');
?>
```

- (Optional) Create a DB user for connection from TestLink. Don't forget to assign a correct rights (at least SELECT, INSERT, UPDATE, DELETE) for the created database. The user must be defined in config_db.inc.php. Otherwise you can use any other user available in MySQL database with correct rights.
- On Linux or UNIX you must change the permissions of the `templates_c` directory to be writable by the webserver. From the TestLink root directory run
- **# chmod 777 gui/templates_c**
- Log into TestLink! Default credentials are:
- user: admin; pass: admin
- Changing this password is a good security practice. TestLink notifies if you don't do it.
- After a successful upgrade you should remove the `<testlinkwebdir>/install/` directory for security reasons.
- The next part involves configuring the installation to work with your specific setup. See configuration section for description of configurable parameter.
- Report any issues or feedback to TestLink Bug tracking system page.

3.4. Upgrading

Major version upgrade: You can upgrade either automatically (via script) or manually. There are typically new functionality and several changes in database against older TL main releases. I.e. you are not able to use directly your original database. (for example 1.6 -> 1.7)

Hot-Fix version is bug fixing release.

3.4.1. Hot-Fix release update

Maintenance (Bug fixing) release is for example 1.6.0 -> 1.6.1. Database schema doesn't changed in this case.

- Backup all files of the previous version in testlink directory.
- Remove the all files from directory.
- Copy a new version to the same directory.
- Copy **config_db.inc.php** and **custom_config.inc.php** file to the new structure and modify configuration parameters acording your previous settings.
- Upgrade of DB is not required.

  Exception: TL main page informs you if small upgrade of DB schema is required (rel. 1.7.1)
- Now, it should work.

3.4.2. Automatic upgrading major version

- Follow/check preinstallation steps. Requirement changes.
- From a web browser run `http://<testlinkwebdir>/install/index.php`
- Choose 'Upgrade Installation' link. Run the scripts until you see that process is finished.
- After a successful upgrade you should remove the `<testlinkwebdir>/install/` directory for security reasons.
- The next part involves configuring the installation to work with your specific setup. See configuration section for description of configurable parameter.
- Report any issues or feedback to TestLink Bug tracking system page.

3.4.3. Manual upgrading

This chapter describe changes in against previous versions. The automatic upgrade is recommended. Use this chapter for a special cases and fiddling config. You can do it of course after a study of changes in database and installation script. Good idea is to compare SQL files for create db tables (your current version and a new one).

3.5. Backward compatibility

3.5.1. Database schema changes

- user password is encrypted (1.5)
- A new tables for SRS feature: requirements, req_coverage, requirement_doc (1.6)
- Attachments (1.7)
- Custom fields (1.7)

3.5.2. Terminology

- Product => Test Plan

- Component, Category => Test Suite

3.5.3. Obsolete functionality from 1.7

- Personal metrics on main page (parameter: MAIN_PAGE_METRICS_ENABLED)

3.5.4. Test Plan relation to Test Project

TL 1.0.4 has not relation Test Plan relation to Test project (Product). The solution from TL 1.6 table include field *TestProjectID* in the Test Plan table. Test Plans could be available over all Test projects (Products). Such Test Plan has *TestProjectID* value = 0.

Configuration within `<testlink_root>/config.inc.php`:

```
$g_show_tp_without_prodid=1;
$g_ui_show_check_filter_tp_by_testproject = 1;.
```

3.5.5. Latin to UTF-8 conversion (upgrade from 1.5 and older)

TestLink 1.6 allows for UTF-8 encoded character rendering, therefore any extended character data that may have snuck into your database and didn't show up in 1.5 may start appearing in 1.6 UI. You can turn UTF-8 support off in testlink by modifying a value in the `<testlinkinstalldir>/config.inc.php` file, but then you will be missing out on the ability to use characters beyond ASCII.

If you have the same problem I did and see lots of extended characters appearing in your data after upgrading to 1.6 and having UTF-8 support turned on, you should read through the following instructions. Be sure to practice this exercise on a test machine before performing on your deployment system.

The instructions will help you clear out any non-ASCII characters from your database and setup your database to support UTF-8.

- First make a backup of your current database using the mysqldump utility.

  ```
  # /usr/bin/mysqldump -u root testlink15 -p > testlink15.backup
  ```

- Now edit testlink15.backup so schema definitions for EACH table has utf8 encoding specified. Change the CHARSET for each table from latin1 to utf8. For example the following line in the definition of 1 table which reads as follows :

  ```
  ENGINE=MyISAM DEFAULT CHARSET=latin1 COMMENT='This table holds the bugs filed for
  each result';
  ```

  should be changed to

  ```
  ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='This table holds the bugs filed for each
  result';
  ```

- Then ran testlink15.backup thru my the perl script below as follows:

  ```
  /replaceScript.pl < testlink15.backup > testlink15.cleaned
  replaceScript.pl is as follows :
  ```

```
#!/usr/bin/perl
while (<>) {
        chomp;
        tr/\000-\177/\040/cs;
        print $_, "\n";
}
```

- Created an empty testlink16 db with utf8 charset as follows:

```
CREATE DATABASE testlink16 CHARACTER SET utf8;
```

- Install the tables into the new database

```
# mysql testlink16 -u root -p < testlink15.cleaned
```

- You can verify your database's "Db characterset" is now set to utf8 by using the following command:

```
login to mysql
use testlink16
mysql> \s
--------------
mysql  Ver 14.7 Distrib 4.1.11, for redhat-linux-gnu (i386)
Connection id:          26
Current database:       testlink15
Current user:           bugz@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         4.1.11
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db     characterset:    utf8
Client characterset:    latin1
Conn.  characterset:    latin1
UNIX socket:            /var/lib/mysql/mysql.sock
Uptime:                 36 min 55 sec
```

- Run the upgrade installation provided by Testlink 1.6.

Other resources:

what the heck is UTF-8 ?

http://www.joelonsoftware.com/articles/Unicode.html

octal table (you can see octal values 000 - 177 are "normal ascii" characters).

The perl script that is provided searches based on octal values.

http://web.cs.mun.ca/~michael/c/ascii-table.html

description of tr perl operation

http://www.unix.org.ua/orelly/perl/learn/ch15_05.htm

3.5.6. Keyword Management

If you don't want to create multiple times the same keyword for the same Test Project:

```
$g_allow_duplicate_keywords=FALSE;
```

4. Configuration

This chapter describe the most important configuration parameters. Additional information are together with parameters definition in configuration files.

4.1. Configuration Files overview

All configuration parameters are inside the file config.inc.php and included files. For this release these are the configuration files:

```
<testlink installation directory>/config.inc.php
<testlink installation directory>/config_db.inc.php
<testlink installation directory>/custom_config.inc.php
<testlink installation directory>/cfg/<bug_tracking_system>.cfg.php
```

config.inc.php - Main configuration file. This file is included into nearly each page. See below for more.

config_db.inc.php - Contains configuration parameters to access the database. This file is created by the installer during the installation or upgrade process. Normally you don't need to change it manually.

custom_config.inc.php – serve for modification of default values of parameters in config.inc.php. The benefit is that your modification is easy to copy during upgrade procedure.

/cfg/<bug_tracker>.cfg.php – set access to database of a bug tracking tool.

4.1.1. custom_config.inc.php

Instead of make changes to **config.inc.php**, we suggest to add your changes to file: **custom_config.inc.php**. This allows you better to save your configuration for the case of update.

Example:

To configure mail server settings, copy following lines into **custom_config.inc.php,** and make changes according to you configuration.

```
$g_tl_admin_email = 'tl_admin@127.0.0.1'; # for problem/error notification
$g_from_email = 'testlink_system@127.0.0.1';  # email sender
$g_return_path_email  = 'no_replay@127.0.0.1';

# Urgent = 1, Not Urgent = 5, Disable = 0
$g_mail_priority = 5;

// SMTP Configuration
$g_smtp_host = 'localhost';  # SMTP server MUST BE configured

// Configure only if SMTP server requires authentication
$g_smtp_username    = '';  # user
$g_smtp_password    = '';  # password
```

4.2. Configuration of Bug Tracker

To enable this feature you need to change a configuration paramenter on the configuration file (custom_config.inc.php). The interface is disabled by default (value 'NO').

The available values are: 'NO', 'BUGZILLA', 'MANTIS', 'JIRA', 'TRACKPLUS', 'EVENTUM' or 'TRAC'.

```
$g_interface_bugs = 'MANTIS';
```

See system requirements chapter for supported versions. The particular BTS configuration file could be:

/cfg/bugzilla.cfg.php

/cfg/eventum.cfg.php

/cfg/jira.cfg.php

/cfg/mantis.cfg.php

/cfg/trac.cfg.php

/cfg/trackplus.cfg.php

Contains configuration parameters to access to particular issue tracking system. You need to edit this file is you want to access issue information from testlink (bugtracking system integration feature). See Appendix for an example of configuration.

4.3. Logging

TestLink has own logging system. You can use it for troubleshooting. A log file is created for each user. Configure the next parameters in config.inc.php file:

LOG LEVEL

Set this to the default level of logging (NONE, ERROR, INFO, DEBUG, EXTENDED). Note that TestLink doesn't verify a size of a created files. I.e. Use DEBUG level only for development or bug investigation to save disc place. ERROR level is recommended for production. It's default settings.

```
$g_log_level='ERROR';
```

LOGGING PATH

The path for the logging of TestLink. E.g. */tmp/* for linux and *c:\temp* for winxp.

```
$g_log_path=TL_ABS_PATH . 'logs' . DS ;
```

PHP logging has Error level by default. We want php errors to show up for users. You can modify it of course. See php.net for more.

```
error_reporting(E_ALL);
```

4.4. Login authentication

TestLink supports two kinds of authentication

- 'MD5' - use encrypted password stored on internal database
- 'LDAP' - use password from LDAP Server

Internal password is default:

```
$g_login_method     = 'MD5';
```

LDAP authentication needs a few more parameters to set:

```
$g_ldap_server              = 'localhost';
$g_ldap_port                = '389';
$g_ldap_root_dn             = 'dc=mycompany,dc=com';
$g_ldap_organization        = '';  # e.g. '(organizationname=*Traffic)'
$g_ldap_uid_field           = 'uid'; # Use 'sAMAccountName' for Active Directory
$g_ldap_bind_dn             = ''; // Left empty if you LDAP server allows anonymous binding
$g_ldap_bind_passwd         = ''; // Left empty if you LDAP server allows anonymous binding
```

4.5. Configuration of global functionality

You can configure the next parameters in config.inc.php file.

TL_IMPORT_LIMIT is maximum upload file size in bytes. Default is 204800. You could increase this value if you import a bigger file. There is also parameter **TL_IMPORT_ROW_MAX** for maximal size of one line of exported file. The value 10000 characters should be enough.

```
define('TL_IMPORT_LIMIT', '204800');
define('TL_IMPORT_ROW_MAX', '10000');
```

4.5.1. Printing

The next strings are used in front page of printed document. Left blank to disable.

This constants must be changed in config.inc.php (not in custom_config.inc.php) for historical reason.

TL_DOC_COMPANY, TL_DOC_COPYRIGHT, TL_DOC_CONFIDENT, TL_DOC_COMPANY_LOGO

```
define('TL_DOC_COMPANY', "Testlink Comunity");
define('TL_DOC_COMPANY_LOGO', '<img alt="TestLink logo" title="Our Logo"
      src="%BASE_HREF%' . TL_THEME_IMG_DIR . '/company_logo.png" />');
define('TL_DOC_COPYRIGHT', '(c) 2003 Testlink Comunity');
define('TL_DOC_CONFIDENT', 'You must share this document');
```

4.5.2. Charset

TestLink supports UTF-8 by default.

```
$g_defaultCharset =  'UTF-8'
```

ISO-8859-1 is possible to configure for backward compatibility. Advanced users also can use another set. We strongly recommend to use unicode character set (UTF-8).

DB_SUPPORTS_UTF8 = [TRUE/FALSE]

Set this to FALSE for MySQL-Versions prior to 4.1 (no utf8 support), so all pages have charset ISO-8859-1 and data will be stored with charset latin1 into the db. For MySQL-Versions >= 4.1 set it to TRUE to enable full UTF-8 support in pages and also data will be stored with charset utf8 into the db.

TL_TPL_CHARSET

This defines a correct html charset. All languages could leave this option unchanged. Modify it only if you know what you do.

```
define('TL_TPL_CHARSET', $g_defaultCharset);
```

4.5.3. Test Case Generation from Requirement

One of the unique features of Test Link is Requirement Management. After creating the Software Requirements Specifications (SRS), and populating it with requirements you can choose to create test cases for every requirement (Component and Category are also created).

Using the configuration object: `$g_req_cfg`, you can configure :

- Name to give to the Test Suite created/used for generated Test Cases:

```
$g_req_cfg->default_testsuite_name="Component Created by Requirement - Auto";
```

- Test Suite's Scope:

```
$g_req_cfg->testsuite_details="Component/Category/Test Cases generated from
Requirements";
```

For the Test Suite name you can configure the following options:

- $g_req_cfg->use_req_spec_as_testsuite_name=TRUE;

  Then REQuirement Specification Title is used a Test Suite name.

- $g_req_cfg->use_req_spec_as_testsuite_name=FALSE;

  Then $g_req_cfg->use_req_spec_as_testsuite_name is used a Test Suite name.

4.5.4. Duplicate names for Test Projects, Test Suites and Test Cases

As you know, is possible to create one of this objects (Test Projects, Test Suites and Test Cases) doing a copy of an existing one.

You can configure how to proceed when the copy is done:

if you set `$g_check_names_for_duplicates=TRUE` then the following checks will be done:

1. *Test Project* name is unique
2. *Test Suite* Name inside *Test Project* is unique

3. *Test Case* Name inside Category is unique

One you have set $g_check_names_for_duplicates=TRUE, you can configure how to proceed, if a duplicate name is found, using `$g_action_on_duplicate_name`.

The options are:

- `'allow_repeat'` : allow the name to be repeated (backward compatibility with version 1.0.4 and 1.5.x)
- `'generate_new'` : generate a new name using the value of $g_prefix_name_for_copy and the original object name.
- `'block'` : return with an error .

Example of formatting:

```
$g_action_on_duplicate_name='allow_repeat';
$g_prefix_name_for_copy= strftime("%Y%m%d-%H:%M:%S", time());
```

4.5.5. Filtering Test Plans by Test Project

As stated before the default behavour is to filter Test Plan by Test Project. Using the following configuration parameter:

```
$g_ui_show_check_filter_tp_by_testproject = TRUE;
```

Allow the user, through the user interface , to enable/disable test plan filter by Test Project. A check box is displayed over the test plan combo box. Force Test Plan filtering, without any user possibility to change it.

```
$g_ui_show_check_filter_tp_by_testproject = FALSE;
```

4.5.6. Test Plan relation to Test Project

Starting with version 1.6 when you create a Test Plan, it's associated to the current selected Test Project as default. This is means you can filter Test Plans by Test Project.

Before Teslink 1.6 the Test Plans where not associated to an specific Test Project. When upgrading from 1.5.x to 1.6, it's not possible for the installer to know to which Test Project relates ogni test plan, then Test Project ID is set to 0. This results in a situation where you find you can't see any of your old Test Plans !!!. To solve this problem the following configuration parameter was added:

```
$g_show_tp_without_prodid=TRUE;
```

You can also via DB administration assign this relation manual and use this feature for data from previous version.

4.5.7. Add a new type of Test results on execution page

You will need to work on following files (all paths are relative to installation dir):

- cfg/const.inc.php

- custom_config.inc.php <-- create it if do not exist yet
- locale/en_GB/custom_strings.txt <-- create it instead of editing strings.txt
- gui/themes/theme_m2/css/testlink.css

1. Open cfg/const.inc.php and search for: $g_tc_status
2. Copy following lines into custom_config.inc.php:

```
$g_tc_status = array (
      "failed"        => 'f',
      "blocked"       => 'b',
      "passed"        => 'p',
      "not_run"       => 'n',
      "not_available" => 'x',
      "unknown"       => 'u',
      "all"           => 'all'
);
...
$g_tc_status_css = array_flip($g_tc_status);
...
$g_tc_status_verbose_labels = array(
            "all"      => "test_status_all_status",
      "not_run"  => "test_status_not_run",
      "passed"   => "test_status_passed",
      "failed"   => "test_status_failed",
      "blocked"  => "test_status_blocked",
      "not_available" => "test_status_not_available",
      "unknown"       => "test_status_unknown"
);
...
$g_tc_status_for_ui = array(
      "passed"  => "test_status_passed",
      "failed"  => "test_status_failed",
      "blocked" => "test_status_blocked"
);

// radio button selected by default
$g_tc_status_for_ui_default="blocked";
```

3. Add new statuses and save:

   tcstatus_1 -> code q

   tcstatus_2 -> code w

4. custom_config.inc.php will be:

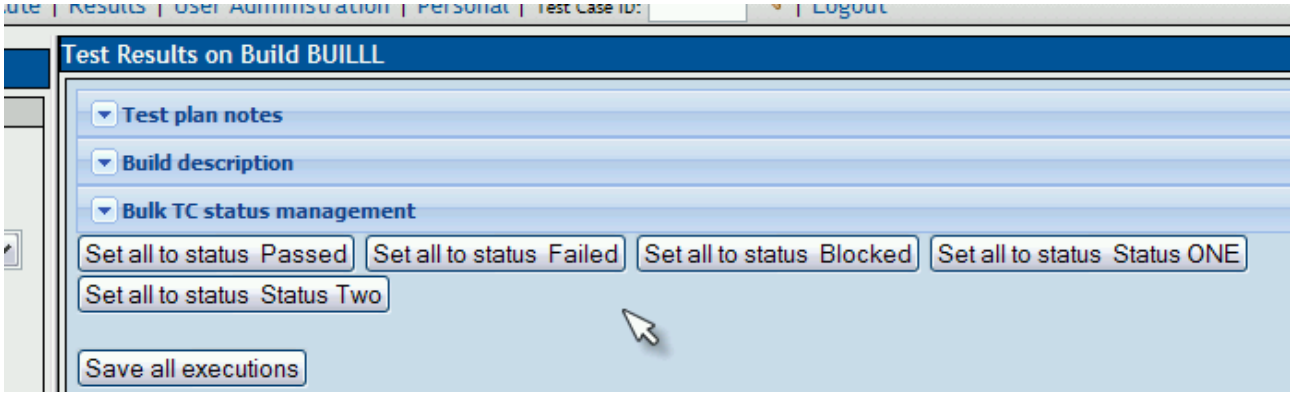
```
$g_tc_status = array (
      "failed"        => 'f',
      "blocked"       => 'b',
      "passed"        => 'p',
      "not_run"       => 'n',
      "not_available" => 'x',
      "unknown"       => 'u',
      "all"           => 'all',
      "tcstatus_1"    => 'q',
```

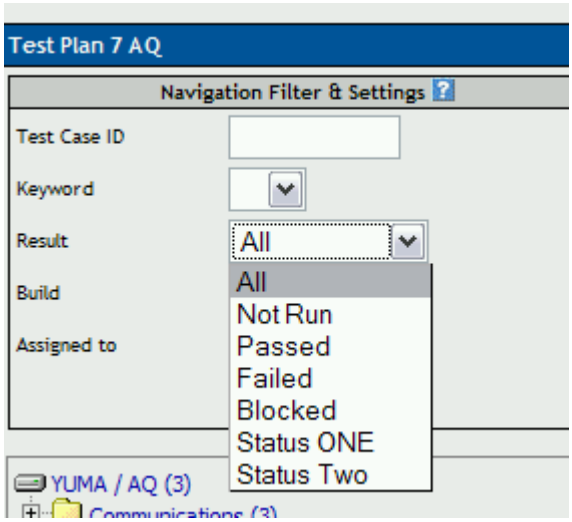
```
      "tcstatus_2"    => 'w'
);
$g_tc_status_css = array_flip($g_tc_status);
$g_tc_status_verbose_labels = array(
  "all"      => "test_status_all_status",
      "not_run"  => "test_status_not_run",
      "passed"   => "test_status_passed",
      "failed"   => "test_status_failed",
      "blocked"  => "test_status_blocked",
      "not_available" => "test_status_not_available",
      "unknown"       => "test_status_unknown",
      "tcstatus_1" => "test_status_new_one",
      "tcstatus_2" => "test_status_new_two"
);
$g_tc_status_for_ui = array(
      "passed"  => "test_status_passed",
      "failed"  => "test_status_failed",
      "blocked" => "test_status_blocked",
      "tcstatus_1" => "test_status_new_one",
      "tcstatus_2" => "test_status_new_two"
);
// radio button selected by default
$g_tc_status_for_ui_default="blocked";
```

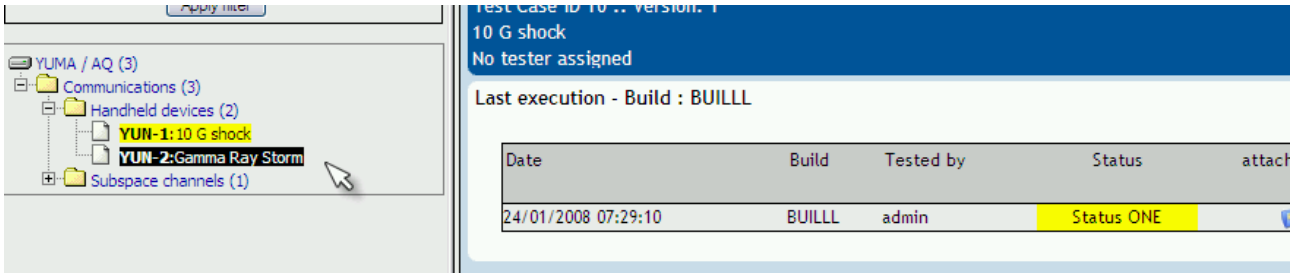
5. Modify css if you want new colors.

```
.tcstatus_1, div.tcstatus_1 {
      color:                black;
      background:           yellow;
}

.tcstatus_2, div.tcstatus_2 {
      color:                black;
      background:           orange;
}

div.tcstatus_1, div.tcstatus_2 {
      margin:               8px;
      padding:              6px;
      text-align: center;
}
```

Effect at user interface level will be







4.6. GUI Customization

You can configure the next parameters in config file:

TL_TREE_KIND

This parameter also is used to configure tree menu Component used in TestLink. Possible values are 'LAYERSMENU', 'DTREE', 'JTREE'. LAYERSMENU is default value. The component JTREE has the best performance. The two others have the ability to remember the last position in addition.

$g_fckeditor_toolbar

fckeditor Toolbar definition. You can modify fckeditor toolbar content. See fckeditor homepage for more information about this Component.

Instead of hard coding attributes of html inputs, like maxlength and size, we have code it on:

```
<TL_INSTALL_DIR>/gui/templates/input_dimensions.conf
```

4.6.1. Cascading Style Sheet

You can change TestLink appearance writing you own CSS (Cascading Style Sheet) files.

You have to change the following constants:

```
define('TL_LOGIN_CSS','gui/css/tl_login.css'); - All Login/Logout pages CSS
```

```
define('TL_TESTLINK_CSS','gui/css/testlink.css'); - Main CSS
define('TL_DOC_BASIC_CSS','gui/css/tl_doc_basic.css'); - Used in Reports
```

Important: paths to CSS are relative to the <testlink installation directory>

If you want to use your own CSS files we suggest you to proceed as follow:

1. create a new directory inside the gui directory, example gui/css/*my_css/*
2. copy the testlink original files to the new directory (you can change the names if you want)
3. modify them at your will
4. edit config.inc.php

```
// Original configuration
//define('TL_LOGIN_CSS','gui/css/tl_login.css');
//define('TL_TESTLINK_CSS','gui/css/testlink.css');
//define('TL_DOC_BASIC_CSS','gui/css/tl_doc_basic.css'); define('TL_LOGIN_CSS','gui/
css/my_css/tl_login_acqua.css');
define('TL_TESTLINK_CSS','gui/css/my_css/testlink_acqua.css');
define('TL_DOC_BASIC_CSS','gui/css/my_css/tl_doc_basic.css');
```

4.6.2. Logo

You can set own logo instead of TestLink default:

```
$g_logo_login_page='<img alt="TestLink" title="TestLink" src="' .
                    TL_THEME_IMG_DIR . '/company_logo.png" />';

$g_logo_navbar= '<img alt="TestLink" title="TestLink" src="' .
                TL_THEME_IMG_DIR . '/company_logo.png" />';
```

4.6.3. Define HTML text editor

Text data editing is solved via javascript editor with toolbar over text area. FCKeditor is used by default as full featured component.

You can define using another editor or just simple text without formating:

```
// 'fckeditor'
// 'tinymce'
// 'none' -> use plan html textarea input field

$g_gui->webeditor='fckeditor';
```

4.6.4. Using Your own Smarty templates (GUI definition)

If You want to test a different solution for the user interface, you can develop your own Smarty Templates. At the time of this writting we have defined the following configuration array:

`$g_tpl`

with the following entries:

- $g_tpl['tcView']

- $g_tpl['tcSearchView']
- $g_tpl['tcEdit']
- $g_tpl['tcNew']
- $g_tpl['execSetResults']

This allows you to create templates with different names that the original Testlink, without the risk to overwrite them, during the next upgrade.

Note: Not all TestLink pages are ready for this kind of configuration.

The standard configuration:

```
$g_tpl['tcView'] = "tcView.tpl";
$g_tpl['tcSearchView'] = "tcSearchView.tpl";
$g_tpl['tcEdit'] = "tcEdit.tpl";
$g_tpl['tcNew'] = "tcNew.tpl";
$g_tpl['execSetResults'] = "execSetResults.tpl";
```

4.7. Attachments

Attachment feature could be enabled (TRUE) /disabled (FALSE):

```
$g_attachments->enabled = TRUE;
```

The type of the repository can be database or filesystem:

- TL_REPOSITORY_TYPE_DB => database
- TL_REPOSITORY_TYPE_FS => filesystem

```
$g_repositoryType = TL_REPOSITORY_TYPE_FS;
```

TL_REPOSITORY_TYPE_FS: the where the filesystem repository should be located

```
$g_repositoryPath = TL_ABS_PATH . "upload_area" . DS;
```

We recommend to change the directory for security reason

Compression used within the repository

- TL_REPOSITORY_COMPRESSIONTYPE_NONE => no compression
- TL_REPOSITORY_COMPRESSIONTYPE_GZIP => gzip compression

```
 $g_repositoryCompressionType = TL_REPOSITORY_COMPRESSIONTYPE_NONE;
```

The maximum allowed file size for each repository entry, default 1MB.

```
define("TL_REPOSITORY_MAXFILESIZE_MB", 1);
```

Also check your PHP settings (default is usually 2MBs)

Users should add a title for the attachment. You can allow left it empty (FALSE). Default is TRUE. The actions for validation (TRUE):

- 'none' - just write on db an empty title
- 'use_filename' - use filename as title

```
$g_attachments->allow_empty_title = TRUE;
$g_attachments->action_on_save_empty_title='none';
```

Title is used as link description for download if title is empty, what the system has to do when displaying:

- 'show_icon' -> the $g_attachments->access_icon will be used.
- 'show_label' -> the value of $g_attachments->access_string will be used .

```
$g_attachments->action_on_display_empty_title='show_icon';
$g_attachments->access_icon='<img src="' . TL_THEME_IMG_DIR . '/new_f2_16.png"
style="border:none">';
$g_attachments->access_string="[*]";
```

You can set own display order of uploaded files.

```
$g_attachments->order_by=" ORDER BY date_added DESC ";
```

4.8. Requirements

Requirement functionality could be enabled / disabled on Test Project level (not via configuration).

Test Case could be generate from Requirements. TestLink will create such Test cases into special Test Suite. You can define that the related SRS title is used (TRUE):

- use_req_spec_as_testsuite_name

FALSE -> test cases are created and assigned to a test suite

with name $g_req_cfg->default_testsuite_name

TRUE -> REQuirement Specification Title is used as testsuite name

*/

```
$g_req_cfg->use_req_spec_as_testsuite_name = TRUE;
```

The next test suite title is used if you set the previous parameter as FALSE :

```
$g_req_cfg->default_testsuite_name = "Test suite created by Requirement - Auto";
```

The next default text will be added here

```
$g_req_cfg->testsuite_details = "<b>Test suite/Test Cases generated from Requirements</b>";
$g_req_cfg->testcase_summary_prefix = "<b>Test Case generated from Requirement</b><br>";
```

Requirement identification string must be unique:

- true : you want req_doc_id UNIQUE IN THE WHOLE DB (system_wide)
- false: you want req_doc_id UNIQUE INSIDE a SRS

```
$g_req_cfg->reqdoc_id->is_system_wide=false;
```

5. Localization

TestLink supports localization of texts, date and time. There is default value in configuration, but each user can set own language. Language code is according to common standards.

```
$g_default_language = 'en_GB';
```

5.1. String localization

A directory exists for every localization, with a standard **`strings.txt`** file inside.

```
<TL_INSTALL_DIR>/locale/de_DE/strings.txt
<TL_INSTALL_DIR>/locale/de_DE/custom_strings.txt
<TL_INSTALL_DIR>/locale/en_GB/strings.txt
...
```

To want to change some of the original translations without changing the provided with the original file provided, you can use **custom_strings.txt.** You need to place this file in the corresponding localization directory, and use the same format and rules used in the original `strings.txt`. If can redefine a value present on `strings.txt`, without need of commenting it on the original file.

Instruction and help pages has own location: `<testlink_root>/gui/help/<language>`.

5.1.1. Date and Time Localization

For every defined locale, you can set the format for date and time presentation. This is configured using the following associative arrays: `$g_locales_date_format` and `$g_locales_timestamp_format`.

At time of this writting the configuration is :

```
$g_locales_date_format = array(
                'en_GB' => "%d/%m/%Y", 'it_IT' => "%d/%m/%Y",
                'es_AR' => "%d/%m/%Y", 'es_ES' => "%d/%m/%Y",
                'de_DE' => "%d.%m.%Y", 'fr_FR' => "%d/%m/%Y",
                'pt_BR' => "%d/%m/%Y" );
$g_locales_timestamp_format = array(
                'en_GB' => "%d/%m/%Y %H:%M:%S",
                'it_IT' => "%d/%m/%Y %H:%M:%S",
                'es_AR' => "%d/%m/%Y %H:%M:%S",
                'es_ES' => "%d/%m/%Y %H:%M:%S",
                'de_DE' => "%d.%m.%Y %H:%M:%S",
                'fr_FR' => "%d/%m/%Y %H:%M:%S",
                'pt_BR' => "%d/%m/%Y %H:%M:%S", );
```

If there is no entry in the previous arrays, the value of the following configuration variables will be used: `$g_date_format` and `$g_timestamp_format`.

Example of formatting:

```
$g_date_format ="%d/%m/%Y";
```

```
$g_timestamp_format = "%d/%m/%Y %H:%M:%S";
```

6. Performance

There are two places where could a problem happen for a large amount of data:

- test case tree menu
- test report with results for particular builds

6.1. Tree menu

There are three component, that you can set to render tree menu of test cases. Default

You can receive:

```
Fatal error: Allowed memory size of 8388608 bytes exhausted (tried to allocate 1328642
bytes)
```

The fast solution is to use the simplest tree menu component “jtree” instead of default “phplayermenu”.

7. FAQ

There are listed often problems. Please, check also TestLink forum.

I upgraded from older version and I cannot login.

Your original database should be in different charset. The default from 1.6 version is UTF-8. Try to switch DB_SUPPORTS_UTF8 to FALSE in config.inc.php.

Smarty error is shown instead of login page.

```
Smarty::include(C:\Inetpub\wwwroot\testlink\gui\templates_c\\%%6A^6A5^6A537DD8%
%login.tpl.php) [function.Smarty-include]: failed to open stream: No such file or
directory in C:\Inetpub\wwwroot\testlink\third_party\smarty\Smarty.class.php on line
1247
```

Linux/unix users: Verify if write permissions are for temp directory (default: `<testlink_root>/gui/template_c/`). Fix by command

```
# chmod a+w <testlink_root>/gui/template_c
```

IIS users: Give the iis_user write access to the template_c directory.

Does Testlink support Secured HTTPS connection?

Yes, it's settings of your web server.

lang_api.php Error is shown instead of login page.

```
[Fri Nov 02] [error] [client xxx.xxx.xxx.xxx] PHP Fatal error: Call to undefined
function iconv() in /home/qa/site/lib/functions/lang_api.php on line 54
```

Note to Windows® Users:
In order to enable this module on a Windows® environment, you need to put a DLL file named iconv.dll or iconv-1.3.dll (prior to 4.2.1) which is bundled with the PHP/Win32 binary package into a directory specified by the PATH environment variable or one of the system directories of your Windows® installation.

This module is part of PHP as of PHP 5 thus iconv.dll and php_iconv.dll is not needed anymore.

How to upload images into text?

http://www.teamst.org/index.php?option=com_content&task=view&id=43&Itemid=2

Allocated memory problem for tree menu

```
Fatal error: Allowed memory size of 8388608 bytes exhausted (tried to allocate 16 bytes) in
D:\wamp\www\testlink173\third_party\phplayersmenu\lib\layersmenu-common.inc.php on line 795
```

phplayersmenu is resource expensive. Switch to another tree component jtree or dtree. Add into custom_config.inc.php:

```
$g_tree_type='JTREE';
```

Appendix A: Setup Mantis bugtracking system integration

A.I Overview

The integration between TestLink and a Bug Tracking System (BTS) has the following characteristics:

- All communication between Test Link and the BTS is done through database tables.
- TestLink (at the time of this writing) is neither able to send data to the BTS, either able to receive data from the BTS, in the traditional model of function call.

After all the configuration is up and running, from a TestLink user point of view the process will be:

1. While executing a test, it fails.
2. User saves execution result.
3. User clicks on link that opens BTS web page used for issue reporting.
4. After issue reporting, user has to take note of issue ID assigned by BTS, to input it into TestLink.
5. User returns to TestLink test execution page, and writes the issue ID in the bug input.
6. After user saves the execution, TestLink will display data taken from the BTS database.

A.II Mantis DB Configuration

Edit file **<your TestLink main directory>/cfg/mantis.cfg.php**.

Environment example: TestLink and Mantis installed on the same web server

Mantis URL	http://calypso/mantis
Test Link URL	http://calypso/testlink
Mantis Database name	mantis_bt
MySQL user/password to access Mantis DB	mantis_bt_user/mantis_bt_password

Anonymous login into mantis has to be turned on. A mantis user with viewer rights to all public projects, must be created. (anonymous account). Change/add following lines in your mantis config_inc.php (replace **dummy** with the anonymous account you will use)

```
# --- anonymous login -----------
# Allow anonymous login
```

```
$g_allow_anonymous_login = ON;
$g_anonymous_account = 'dummy';
```

A.III Enable BTS integration

Check the following lines from **config.inc.php** .

```
// ----------------------------------------------------------------------------
/** [Bug Tracking systems] */
/**
* TestLink uses bugtracking systems to check if displayed bugs resolved, verified,
* and closed bugs. If they are it will strike through them
*
* NO        : no bug tracking system integration
* BUGZILLA  : edit configuration in TL_ABS_PATH/cfg/bugzilla.cfg.php
* MANTIS    : edit configuration in TL_ABS_PATH/cfg/mantis.cfg.php
...
*/
$g_interface_bugs='NO';
```

Copy it to **custom_config.inc.php and** change line:

```
$g_interface_bugs='NO';
```

Final result:

```
$g_interface_bugs='MANTIS';
```

A.IV Check interface

After your configuration is OK, you will find the icon to add bugs in the execute screen.

Several checks are done when you try to add the bug:

- Bug ID is present on BTS ?
- Bug ID format is valid ?

Appendix B: Revision History

#	Description	Date	Author
1.0	Initial creation of the document in DocXML	2005/03/12	A. Morsing
1.1	Corrected title, updated structure and added new sections.	2005/04/12	M. Havlat
1.2	Added some words for MySQL 4.1, UTF8 support	2005/06/27	A. Morsing

#	Description	Date	Author
1.3	Updated automatic installation part	2005/09/12	F. Mancardi
1.4	Updated for TL 1.6.; added configuration parameters; restructured (created pre-installation steps section); corrected layout; added phpMyAdmin steps description	2005/09/13	M. Havlat
2.0	Converted to OO2 format; added DB Charset update explanation from Kevin	2005/12/04	M. Havlat
2.1	Corrected layout for export to HTML and PDF	2005/12/11	M. Havlat
2.2	Some small changes	2005/12/17	A. Morsing
2.3	Minor layout and grammar update	2006/02/14	M. Havlat
2.4	Updated for TL 1.7	2006/11/17	M. Havlát
2.5	Updated for TL 1.7; restructured; merged BTS case; layout update (prepare for 1,7,0 release)	2007/09/13	M. Havlát
2.6	Added several new parameters for 1.7, updated styles, configuration divided into logical chapters	2008/01/02	M. Havlát
2.7	Fixed: 0001347 , 0001284 , 0001331 New sections: Add a new type of Test results on execution page (drafted by Francisco), Define HTML text editor	2008/02/02	M. Havlát